Socio-Technical Systems Models of the Decision Chain in UAV Operations

Dr. Robert J. Houghton [1], Prof. C. Baber [1] and Mr Steve Harmer [2]
[1] EECE, The University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK
[2] MBDA, Golf Course Lane, Filton, Bristol, BS34 7QW, UK

R.J.Houghton@bham.ac.uk / C.Baber@bham.ac.uk

ABSTRACT

By defining the primary functions involved in conducting Unmanned Vehicle missions, it is possible to evaluate different configurations of organisations to support these functions. These could vary, for instance, from a linear chain to a highly connected network. Each structure has different advantages and disadvantages, in terms of information flow, and each poses different costs in terms of organisational behaviour and management. By representing the alternative structures in terms of use-case diagrams (which are aligned to the NATO Architecture Framework), and by modelling the activity described in these views, it is possible to compare and contrast structures and explore their resilience. Modelling is performed in two ways. First, Event Calculus is used to consider the binding between tasks in order to explore unintended consequences of particular configurations. This helps demonstrate how some configurations can lead to potential problems and errors in information flow. Second, a form of dynamic use-case modelling is used to show how binding of tasks to actors can lead to changes in the structure of the organisation, particularly when the availability of actors becomes compromised or when tasks become blocked; thus, providing a novel approach to consider resilience in these networks.

1.0 INTRODUCTION

The purpose of the present paper is to describe two different, but complimentary, approaches to analysing military socio-technical systems (that is, systems comprising of people and technology) that we feel are most useful to promoting the design of safe and resilient systems for modern warfighting. Another motivation for the present work was to pilot these approaches prior to further work that will include an elicitation stage; as such the models considered herein should be regarded as 'strawman' that are not necessarily accurate to real operations. The present concept assumes a series of activities in military operations from a person on the ground calling in a UAV to its dispatch; these activities include planning, coordination and seeking legal clearance. One approach would be to see this as a linear chain of events. However, increasingly uncertainty in enemy identification and movement, together with rules of engagement intended to limit "blue-on-blue" and collateral damage may mean that linear chains are not always appropriate, particularly when the tempo of change in ground truth is particularly rapid. However, there is a tension between safety and assurance and flexibility that must be balanced. In the present work therefore we model the same concept of operations in two different ways that embody different paradigms; linear workflow and distributed operations aided by computer support. Both models are produced using a formalism called the Event Calculus that allows reasoning about events over time using logical propositions. We then consider how these two different approaches could be manned using dynamic use cases.

Report Documentation Page

Form Approved OMB No. 0704-0188

1. REPORT DATE **OCT 2009**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **Socio-Technical Systems Models of the Decision Chain in UAV Operations**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **EECE, The University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

See also ADA562563. RTO-MP-MSG-069 Current Uses of M&S Covering Support to Operations, Human Behaviour Representation, Irregular Warfare, Defence against Terrorism and Coalition Tactical Force Integration (Utilisation actuelle M&S couvrant le soutien aux operations, la representation du comportement humain, la guerre asymetrique, la defense contre le terrorisme et l'integration d'une force tactique de coalition). Proceedings of the NATO RTO Modelling and Simulation Group Symposium held in Brussels, Belgium on 15 and 16 October 2009., The original document contains color images.

14. ABSTRACT

By defining the primary functions involved in conducting Unmanned Vehicle missions, it is possible to evaluate different configurations of organisations to support these functions. These could vary, for instance, from a linear chain to a highly connected network. Each structure has different advantages and disadvantages, in terms of information flow, and each poses different costs in terms of organisational behaviour and management. By representing the alternative structures in terms of use-case diagrams (which are aligned to the NATO Architecture Framework), and by modelling the activity described in these views, it is possible to compare and contrast structures and explore their resilience. Modelling is performed in two ways. First, Event Calculus is used to consider the binding between tasks in order to explore unintended consequences of particular configurations. This helps demonstrate how some configurations can lead to potential problems and errors in information flow. Second, a form of dynamic use-case modelling is used to show how binding of tasks to actors can lead to changes in the structure of the organisation, particularly when the availability of actors becomes compromised or when tasks become blocked; thus, providing a novel approach to consider resilience in these networks.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**18**	

2.0 THE EVENT CALCULUS

The event calculus (EC) is a logic-based formalism for reasoning about events and time [1,2]. Although originally motivated by an interest in formalising the logical representation time and applications in database updating and narrative understanding, this technique has been used since in myriad applications including commonsense reasoning for artificial intelligence [3,4]; making business systems more flexible [5]; and high-level vision for cognitive robotics [6]. We propose it may also be used to represent and model time-varying elements of sociotechnical systems.

The four key elements of the event calculus are sorts, fluents, events and time. Sorts are things in the world that we use to anchor our model upon (e.g., we might talk about a sort called a “target”and a specific type of target might be a “Suspicious vehicle”). Fluents are properties of the the world that have different truth values over time. Events, which occur at different points in time, alter the truth of values of fluents. Thus if a door opening event occurs at a given point in time, the truth value of the fluent “door closed” changes from true to false. We generally assume by default that the ‘commonsense law of intertia’ holds, that is, we assume that the door then remains open until an event occurs that causes it to be closed (or an event occurs that causes a door closing event to occur and so on). Furthermore, unless we know otherwise, we also assume that other events, such as switching on a light-switch, do not affect the state of the door. These assumptions mean that we do not have to write out large lists of all the things events *do not do*, just those things they *do*. Four basic EC predicates set out the formal relationship between fluents, events and time (Table 1). Combined with standard first-order logic, the EC has broad descriptive power not limited to; context-sensitive effects, triggered events, concurrency, indirect effects, nondeterministic effects, continuous change, and the representation of space, mental states, emotions etc. subject to a sufficient axiomisation.

Predicate	**Explanation**
`Happens(`e`,`t`)`	Event *e* happens at time *t*
`HoldsAt(`f`,`t`)`	Fluent *f* is true at time *t*
`Initiates(`e`,`f`,`t`)`	If event *e* occurs at time *t*, fluent *f* will be true after time *t*
`Terminates(`e`,`f`,`t`)`	If event *e* occurs at time *t*, fluent *f* will be false after time *t*

Table 1. Event calculus predicates

In terms of putting a system together, we combine a narrative of events (what events happened when and what the state of the world was at different points, see Figure 1), an axomisation of the domain of interest (what events do) together with the logic underlying the EC itself to produce a model that can be queried as to the truth value of fluents at given timepoints. Since the EC can be implemented using a computer these pieces of information take the form of a computer program that takes the form shown in Figure 2. In the present paper we focus on the ‘Discrete Event Calculus’ which represents time as integer timepoints [3] that we treat as arbitrary epochs.

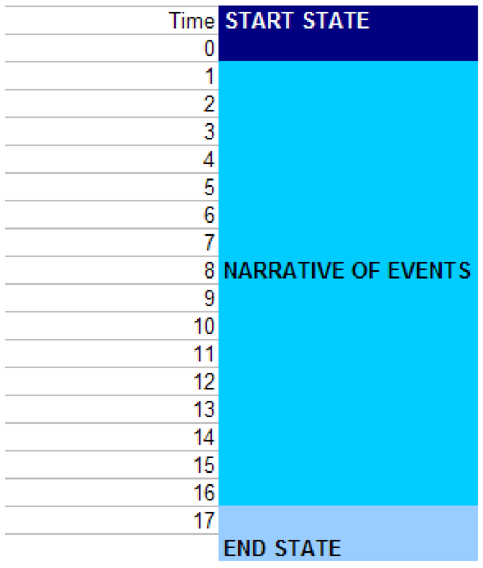


Figure 1. A narrative of events

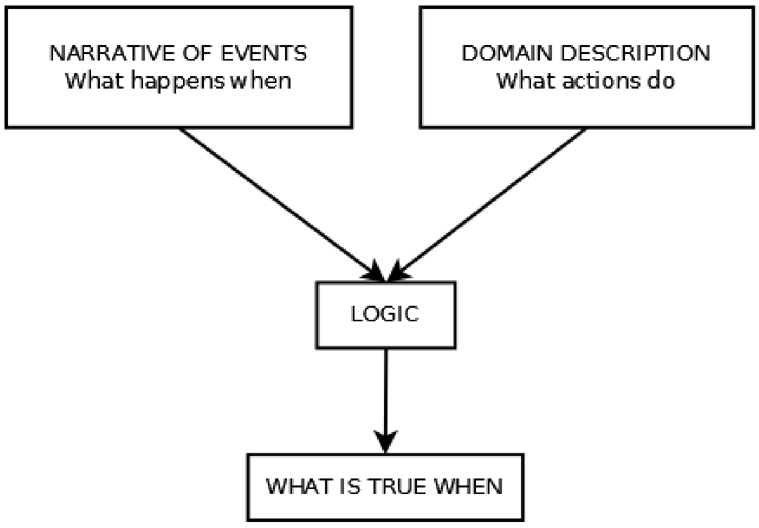


Figure 2. Elements of an event calculus model (adapted from Mueller, 2006)

While such a model would be useful in itself for checking, for example, whether the system as we understand it can be predicted to reach a successful end state (i.e., the truth value of a fluent standing for ultimate success of a mission is true) given a certain narrative of events, the event calculus also allows, by virtue of treating time symmetrically, different types of reasoning to take place (Figure 3).

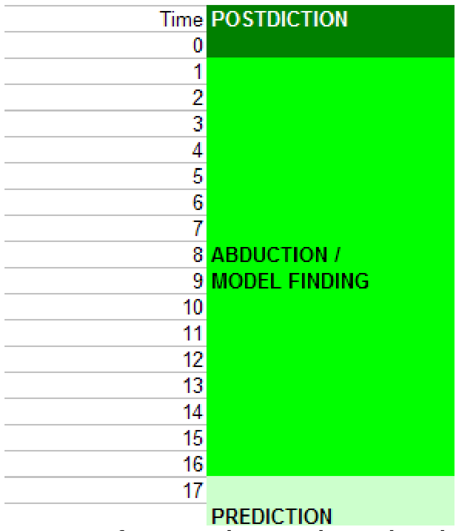


Figure 3. Different types of temporal reasoning using the event calculus

Given a start state and an ordered list of events, we can deduce what the possible end state(s) would be by deduction. Given an end state and a narrative of events we can also work out what the possible start state(s) were. Finally and perhaps most powerfully, given a start state and an end state we can also carry out a form of reasoning termed *abduction*, that is, we can generate a plan or set of plans leading from the start state to the end state. We may also wish to use a partial narrative/set of observations concerning the state of fluents in the world and use the event calculus to generate ways of filling the gaps consistent with those facts. For example, if we know a door was closed and the room was empty at timepoint 1, and that someone was in the room at timepoint 5, then it can be deduced that some point in between, the door was opened to allow someone to enter. If the door was also locked, this would imply that had to be unlocked and imply it turn that the person who unlocked it had a key and so on.

Therefore in terms of modelling and evaluating a system, we have two main approaches that we can employ. We can change the system (as it were) by altering its constituent axioms, and we can change the narrative to test that system across different patterns of events. Indeed, by giving a narrative with a (semantically) negative ending (e.g., the fluent "target reached" is false) we can also generate narratives that expose latent and perhaps complex chains of events that can cause system failure [7]. While this is arguably a reductionist way of representing events and tasks within a system, it does to some extent avoid the worst excesses associated with representing tasks as either hierarchical or ordered in a manner that is brittle in the face of changing demands or elaboration [8,9]. Where events can be fluidly ordered or concurrent, this will be demonstrated, and if events do fall within a strict order, the reasons for this are directly demonstrable within the domain of interest itself (i.e., they result from genuine causal constraints and contingencies) rather than emerging as result of the analyst's need to impose order. Further, with reference to the meanings of the fluents changed by events, events can also be considered as meaningful and are given a specific context relative to what we might call a 'work story' rather than being regarded as generic 'operations per unit time'.

Given the EC has broad powers of representation, in modelling any domain, particularly one as complex as a socio-technical system, it is necessary to decide at what level of analysis it is to be modelled and what sort of ontology should be used. For example, it would be possible (although not necessarily useful in the present case) to represent the system spatially, with entities that occupy particular positions in space. Alternatively, the mental states of a single individual could be represented in great detail. In the present instance we decided the most appropriate level of representation was to consider the decision chain as if it were a workflow model and as if it were distributed system governed by a contract-like protocol in which participants are aware they can only do certain things if specific conditions are met. This is related to extant work in the EC literature for considering protocols in terms of commitments between entities [5,10] and the general area of modelling contracts and legal discourse in logic e.g., [14].

2.1 System as workflow

The first implementation of a work-flow for planning and tasking was based around considering tasks as part of a largely linear workflow as shown in Figure 4. The model was inspired by the author's own observations of command teams at work although it should not be necessarily considered accurate but rather stands as a generic 'strawman' example to illustrate the approach. The purpose of the system is, from the entry point of target identification, to move through various forms of planning (threat assessment, coordination of airspace) and coordination/orgnisation (seeking approvals and clearances) until the point at which a UAV asset can be tasked and dispatched. Thus target identification can be regarded as the start state that initiates the process, and tasking assets can be regarded as the final endstate and goal.

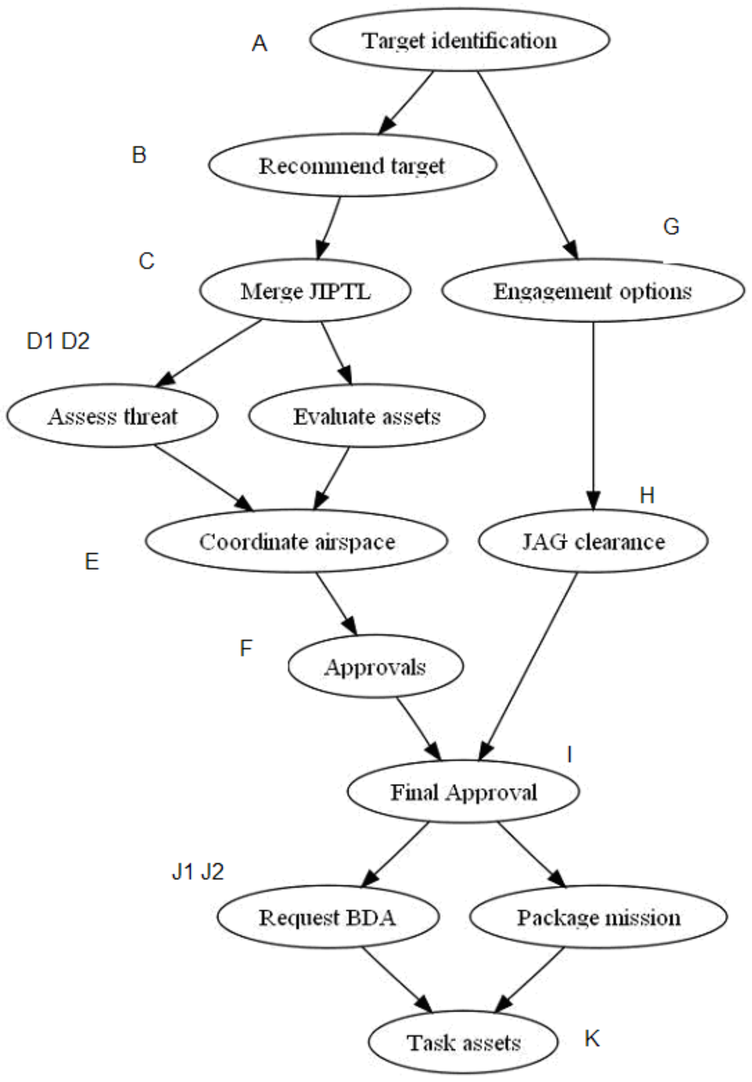


Figure 4. Workflow from target identification to the tasking of assets

This was modelled in DEC (Discrete Event Calculus, see [3]) using an axiomisation suggested by Nihim Kesim Cicekli and Yakup Yildirim [11] that meets the standards put forth Workflow Management Coalition to represent; sequential activity, AND-splits/joins, XOR-splits/joins and iteration [12].

A set of axioms allow for activities to be in a state of being of either `Active` or `Completed` owing to Start and End activity events;

```
Initiates(Start(activity),Active(activity),time).

Terminates(Start(activity),Completed(activity),time).

Initiates(End(activity),Completed(activity),time).

Terminates(End(activity),Active(activity),time).
```

Further axioms implement the logic of workflow, so for example in Figure 4, Activity B (Recommend target) can only occur once activity A (target identification) has Ended and when Activity B is not itself already active[1].

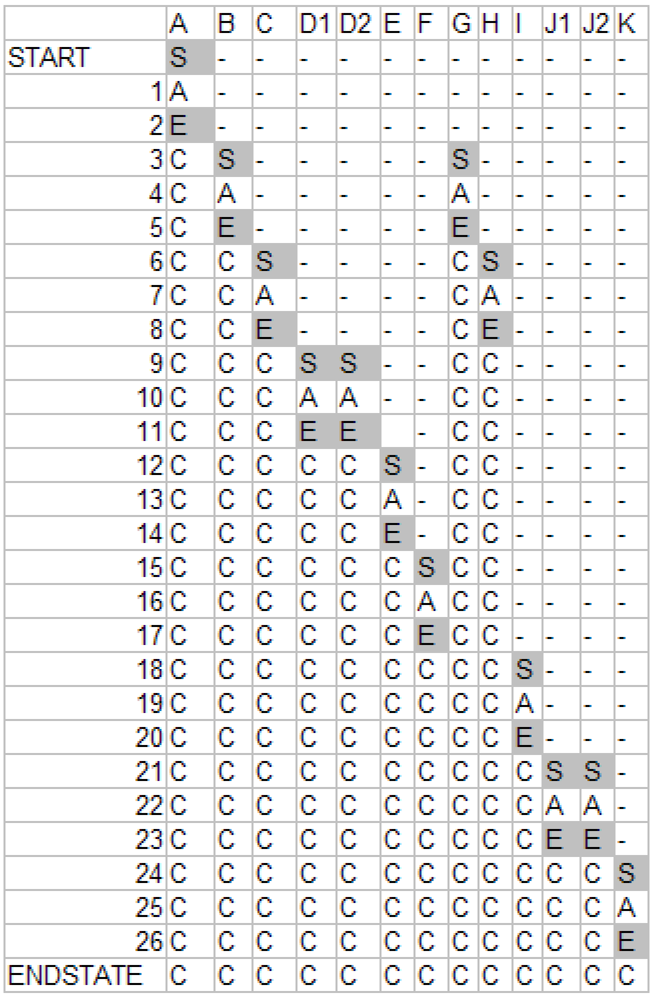
```
Happens(Start(B),time -> !HoldsAt(Active(B),time) &

!HoldsAt(Repeatlock(B),time) &

HoldsAt(Completed(A),time).
```

Once axiomised a model is produced that can then be tested; given a fixed time frame of 26 epochs only a set of models were produced. Given this time limit, the left-hand thread (B→C→D1/D2→E→F→I) is constrained to only 'fit' if D1+D2 are concurrent; meanwhile there other models were possible but this only concerned when activities G (Engagement options) and H (JAG clearance) occurred; the limiting factor being JAG Clearance had to be completed in time to meet the synchronisation point of Final Approval (J) (see Figure 5; the latest that event H could have started would be epoch 14, thus the latest G could end would be epoch 13). Counterfactual models, where for example Assess Threat (D1) took an epoch longer to complete than Evaluate Assets (D2), because the chain as a whole cannot complete in time; or when Final Approval (I) was given without Engagement Options (G) (because it meant in turn JAG clearance could not happen) occurring were ruled out.

[1] The '`Repeatlock`' fluent is used to prevent repeated triggering of B in response to `Completed(A)` being true; it is only allowed to occur once. Mueller (2006, p. 258-259) utilises the expression `!HoldsAt(Completed(A),time-1)` to limit perseveration but this requires that B be triggered immediately after A is completed and *only* immediately after A is completed.



	A	B	C	D1	D2	E	F	G	H	I	J1	J2	K
START	S	-	-	-	-	-	-	-	-	-	-	-	-
1	A	-	-	-	-	-	-	-	-	-	-	-	-
2	E	-	-	-	-	-	-	-	-	-	-	-	-
3	C	S	-	-	-	-	-	S	-	-	-	-	-
4	C	A	-	-	-	-	-	A	-	-	-	-	-
5	C	E	-	-	-	-	-	E	-	-	-	-	-
6	C	C	S	-	-	-	-	C	S	-	-	-	-
7	C	C	A	-	-	-	-	C	A	-	-	-	-
8	C	C	E	-	-	-	-	C	E	-	-	-	-
9	C	C	C	S	S	-	-	C	C	-	-	-	-
10	C	C	C	A	A	-	-	C	C	-	-	-	-
11	C	C	C	E	E		-	C	C	-	-	-	-
12	C	C	C	C	C	S	-	C	C	-	-	-	-
13	C	C	C	C	C	A	-	C	C	-	-	-	-
14	C	C	C	C	C	E	-	C	C	-	-	-	-
15	C	C	C	C	C	C	S	C	C	-	-	-	-
16	C	C	C	C	C	C	A	C	C	-	-	-	-
17	C	C	C	C	C	C	E	C	C	-	-	-	-
18	C	C	C	C	C	C	C	C	C	S	-	-	-
19	C	C	C	C	C	C	C	C	C	A	-	-	-
20	C	C	C	C	C	C	C	C	C	E	-	-	-
21	C	C	C	C	C	C	C	C	C	C	S	S	-
22	C	C	C	C	C	C	C	C	C	C	A	A	-
23	C	C	C	C	C	C	C	C	C	C	E	E	-
24	C	C	C	C	C	C	C	C	C	C	C	C	S
25	C	C	C	C	C	C	C	C	C	C	C	C	A
26	C	C	C	C	C	C	C	C	C	C	C	C	E
ENDSTATE	C	C	C	C	C	C	C	C	C	C	C	C	C

Figure 5. A legal path through the workflow model; S=Start (event), A=Active (fluent), E=End (event), C=Completed (fluent), in the dashed cells the fluents Active and Completed are both false.

When a longer time frame was supplied (30 epochs) a wide range of plausible models were then possible, but clearly delays before synchronisation points propagated forward, thus in domain terms a delay in Coordinating Airspace (E) pushed back the eventual Task Assets (K) to at least a commensurate degree assuming no delays after that point.

This way of modelling a decision chain highlights some of advantages and disadvantages of managing organisations in this way. A signal advantage of the workflow approach is that, for example, we can be assured that assets are not tasked before JAG clearance has been given as a prior synchronisation AND-gate at the point of final approval requires that it has already happened. However, the issue of delays prior to synchronisation points propagating forward illustrates the disadvantages of working in this manner. Thus by choosing an axiomisation we can represent *a way of thinking about things*, in this case, the notion that decision chains should be largely linear and that through AND gating can be synchronised at given decision points (points E, I and K in the above).

2.2 A distributed system

An alternative way of considering (and thus axiomising) a decision chain is in terms of information requirements and the states prior decision reach. This is broadly the notion that Network Enabled Capability/Network Centric Warfare will allow more distributed activities through the shared provision of information and great situational awareness not just of the situation in the field but also of what colleagues (perhaps geographically distributed across the battlefield or maybe even the world) are doing. With such information available to be pulled from and pushed onto the network on demand, there is no longer necessarily the requirement to wait to be passed that information sequentially in the planning process. This can increase efficiency and also allow the system to be more responsive to changing circumstances and perform more flexibly to take advantage of fleeting opportunities. However, there is the risk that such approaches could lead to desynchronisation of the team as a whole and dangerous or unwanted events occurring. This poses a problem for analysis and management: how can we be sure given that activities could occur in myriad orders that we have designed a protocol that will be safe? We suggest one way is to model the protocol using the Event Calculus.

Another way of thinking about this is to consider a system protocol as a contract. As with written contracts, at various stages in the lifespan of contract certain things must be true for it to continue. In the event these things fail to be true, either the contract as a whole fails or certain tasks must be undertaken or re-undertaken in rectification. As well as a statement of what *must* happen, a contract will also describe various things that *must not* happen or else the contract to fail. Changes in these states either allow further actions that were anticipated to occur or trigger specific events in response. Note that owing to the temporal extension of the contract, states may switch back and forth between being true and false during its lifetime. A particular problem in temporally extended contracts is where events and states might interact in unforeseen ways; this may occur because the contract itself is badly structured (this stepping back from the analogy, the decision chain has potential flaws) or because events have not unfolded in the order originally anticipated.

In formalising a decision chain we used various fluents to represent this time not the status of given activities, but rather their outcomes; knowledge of the target's location, JAG approval and so on, has planning been completed and so on. For example the fluent `KnowLocationofTarget` can become true only after target identification:

```
Initiates(Targetidentification(target),KnowLocationTarget(target),time).
```

(i.e., After target identification of a target, it becomes true that we know the location of a target).

In the case of some activities they may alter more than one fluent. For example, in the case of coordinate airspace we use a fluent to record that this has occurred (because it is necessary for packaging the mission) and also that airspace has been deconflicted (which is later necessary at the point of tasking assets).

```
Initiates(Coordinateairspace(target),
Airspacecoordinated(target),time).

Initiates(Coordinateairspace(target), Deconfliction(target),time).
```

Similarly, other activities are possible only when multiple fluents are true; for example; it is only valid to give final approval (making the mission approved) when it is true that approval has been given and JAG clearance issued and we know the location of the target:

```
Happens(Finalapproval(target),time)-> HoldsAt(Approved(target),time) &
HoldsAt(JAGcleared(target),time)& HoldsAt(KnowLocationTarget(target),time)
```

It may at first blush appear curious that we have included the requirement to know the location of the target at such a late state as presumably we would have to know target location in advance of giving initial approval and JAG clearance. The point of its inclusion is that if we force a change in that fluent to false (i.e., we have lost the target), it is clear how the decision chain can shutdown or reiterate to a previous point depending on the circumstances. Final approval could not occur as there would be an inconsistency in narrative between not knowing the location of the target and needing to give final approval which requires knowledge of the location of the target:

```
!HoldsAt(KnowlocationofTarget(Suspicioustruck),5)

Happens(Finalapproval(Suspicioustruck),5)
```

Furthermore by including these indirect dependencies we can ensure the same level of security that all that needs to be decided and known has been decided and known under more fluid orderings of events than a fixed task flow. Similarly, if we recast JAG approval as something that is either true or untrue at any point in the process (rather than making it a specific decision point in a linear chain) then this distributed protocol shows how it would be possible to withdraw approval at any point in the life of mission planning (as for example new information comes in) rather than at a single point. Such an approach would be possible given modern IT and furthermore would arguably be more appropriate to warfighting in highly dynamic circumstances that feature high levels of uncertainty. The point at stake here is that ground truth can change faster than the planning cycle can be completed and executed, it may be necessary to look at ways of reorganising planning.

An EC model can also ‘trap’ dangers inherent in a protocol. While in Figure 4 we see coordinating airspace is a procedural step dependent on the prior assessment of threat and assets, in our information requirements model we had not expressed this, nor had we made it a requirement for command approval; thus it was possible to find a path through the model that allowed a mission to be launched without airspace deconfliction having taken place. However, because the EC is elaboration tolerant, it was straight-forward to add a clause that airspace deconfliction should be in place prior to mission packaging. In terms of constructing a model of a real-life domain, at such points the analyst would return to subject matter experts/observational data/documentary evidence and consider whether the problem lies in their knowledge of the domain or whether a real problem has indeed been identified.

On setting these definitions up such that abduction (plan finding) could take place we found a range of models that complete all the required activities within 20 epochs. The general form of this class of models is shown in Figure 6. It is to be noted that while Figure 4 (workflow) was drawn and then formalised, this task flow was generated by the EC logic program itself and then drawn to permit comparison. It might also be considered slightly misleading in that in contrast to the workflow axiomisation, the event “Request BDA” is in point of fact as dependent upon “Target Identification” (which changes the `KnowLocationOfTarget` fluent from false to true) as it is on the preceeding “Engagement options” event (which changes the fluent `Missionplanned` from false to true).

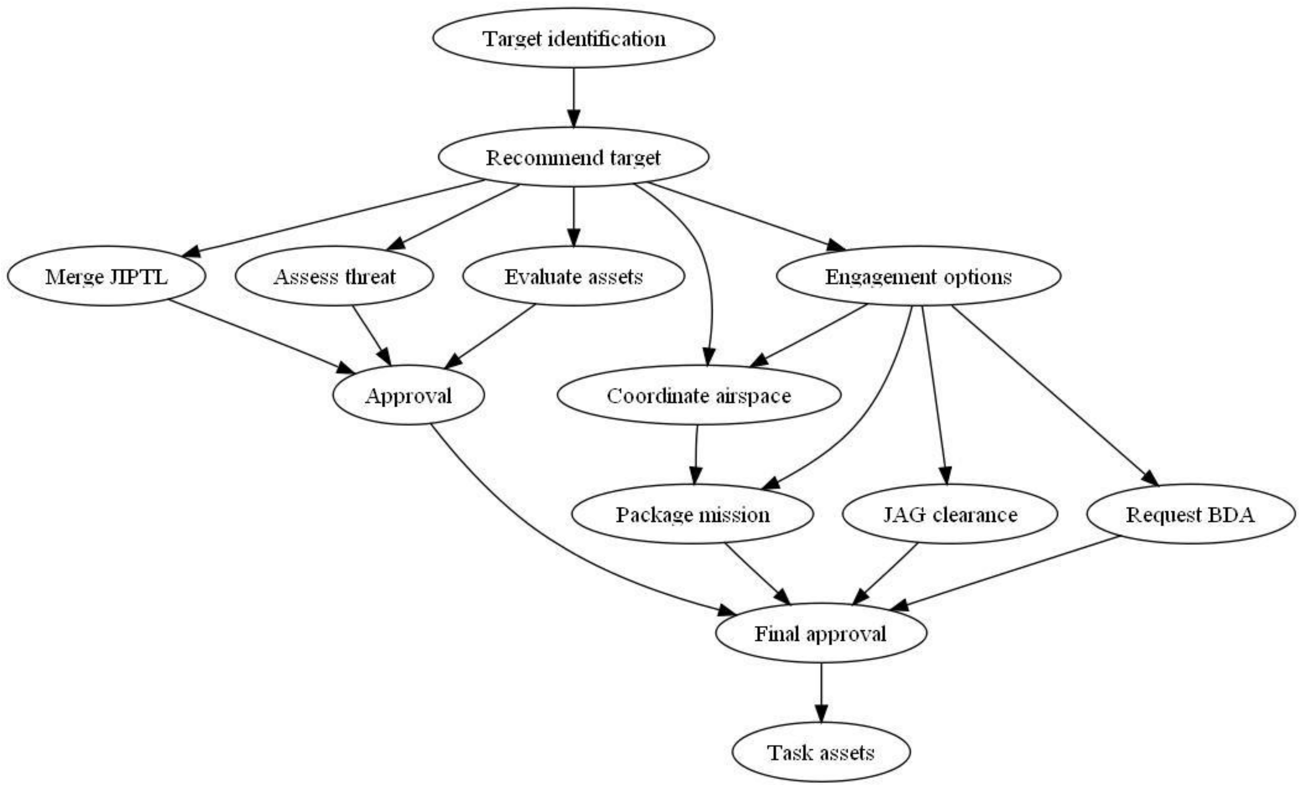
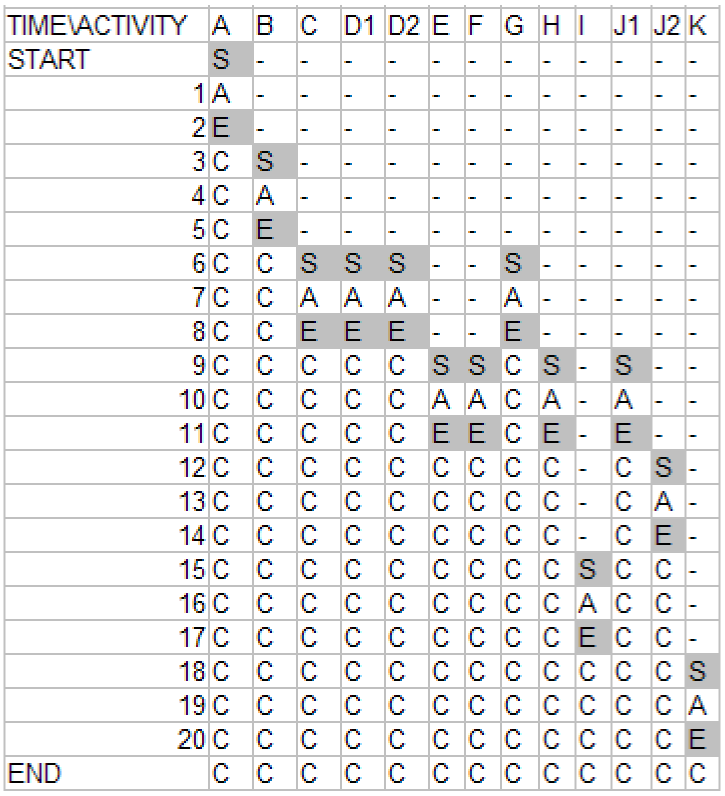


Figure 6. Process generated by EC model on the basis of information requirements

We did not use the same 2-state (active/completed) formalisation for each activity, but when adjusted into this form for post-hoc for comparison, we find Figure 7. Although it was not our intention to necessarily aim to find faster ways the decision chain could complete, owing to greater parallelism this version can complete at least six epochs (arbitrary time units) faster than the workflow form of the same concept of operations. It is also to be noted that, given a sufficiently long time limit, the workflow model in Figure 4 was also one of the alternate models that could be produced; the distributed representation does not necessarily contradict the previous approach, rather it expands upon it and finds wider options on the basis of the information given to it.

From the point of managing this process, we see that the approval and final approval activities have become pivotal synchronisation points, but that further, activities prior to the tasking of mission assets could take place before, rather than after, this point is reached. This could present challenges particularly in terms of commanders becoming situationally aware of the progress of the planning process itself. We assume that technology could have a part to play in this; indeed, the EC simulation we have produced (although not very user friendly at this stage given it consists entirely of textual logical propositions) could in fact form the backbone of a software package to assess in real-time what activities were possible, where bottlenecks were forming and whether (perhaps assuming various typical timings) the planning process can complete on time. We hope to undertake future work to evaluate this possibility of repurposing EC simulations as decision support tools.



TIME\ACTIVITY	A	B	C	D1	D2	E	F	G	H	I	J1	J2	K
START	S	-	-	-	-	-	-	-	-	-	-	-	-
1	A	-	-	-	-	-	-	-	-	-	-	-	-
2	E	-	-	-	-	-	-	-	-	-	-	-	-
3	C	S	-	-	-	-	-	-	-	-	-	-	-
4	C	A	-	-	-	-	-	-	-	-	-	-	-
5	C	E	-	-	-	-	-	-	-	-	-	-	-
6	C	C	S	S	S	-	-	S	-	-	-	-	-
7	C	C	A	A	A	-	-	A	-	-	-	-	-
8	C	C	E	E	E	-	-	E	-	-	-	-	-
9	C	C	C	C	C	S	S	C	S	-	S	-	-
10	C	C	C	C	C	A	A	C	A	-	A	-	-
11	C	C	C	C	C	E	E	C	E	-	E	-	-
12	C	C	C	C	C	C	C	C	C	-	C	S	-
13	C	C	C	C	C	C	C	C	C	-	C	A	-
14	C	C	C	C	C	C	C	C	C	-	C	E	-
15	C	C	C	C	C	C	C	C	C	S	C	C	-
16	C	C	C	C	C	C	C	C	C	A	C	C	-
17	C	C	C	C	C	C	C	C	C	E	C	C	-
18	C	C	C	C	C	C	C	C	C	C	C	C	S
19	C	C	C	C	C	C	C	C	C	C	C	C	A
20	C	C	C	C	C	C	C	C	C	C	C	C	E
END	C	C	C	C	C	C	C	C	C	C	C	C	C

Figure 7. A legal path through the distributed model found by the EC planner carrying out logical abduction. S=Start (event), A=Active (fluent), E=End (event), C=Completed (fluent), in the dashed cells the fluents Active and Completed are both false. Activity coding is per Figure 4.

3.0 USE-CASES

One way of examining the resilience of a task-actor network to disruption is some form of stress testing. Various approaches to the analysis of network resilience have been proposed [15] although there is little consensus at present as to which techniques and measures are most appropriate in given situations. In the present case we are concerned with networks of actors and functions which differ from the sorts of physical networks (e.g., the internet, transport networks) and intelligence networks (e.g., organization of terrorist cells) in that they are in a sense 2nd order representations of a pattern of behavior in a work situation rather than entities in and of themselves. Therefore we first staffed the tasks described in Section 2 in a generic manner (in the absence of more detailed real-world data identifying specific individuals) considering actors: Recce, Ops, Plans, Intel, JAG and Command. The pattern of staffing is shown in Figure 8. Recce is the agent outside the command group itself who makes the initial identification of the target.

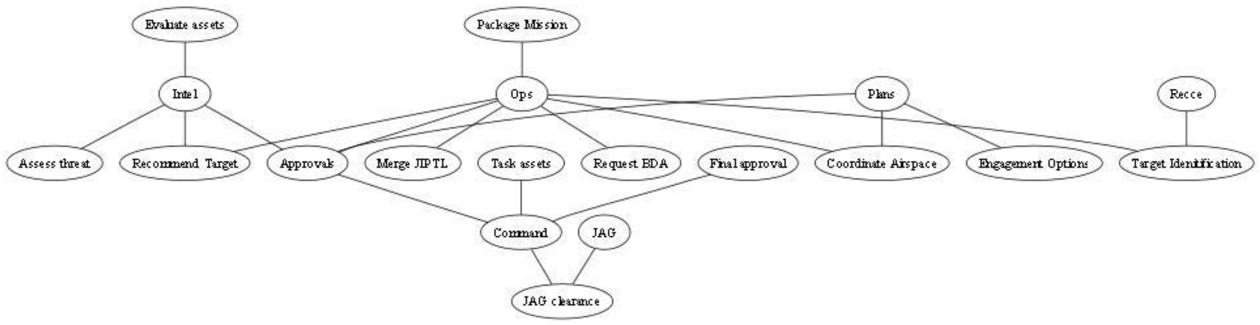


Figure 8. Use-case (actors x tasks)

A pattern of interactions between actors by shared tasks is shown in Figure 9. For example, the Recce (who call in the target) are connected indirectly to other actors via Ops who receive the call and begin the planning process.

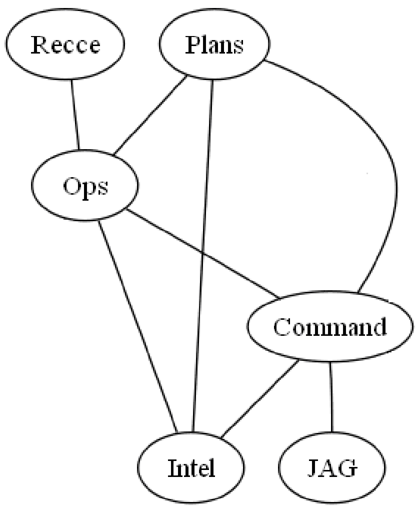


Figure 9. Collaboration graph

We then submit this network to the removal of actor nodes. Because the networks represent what work actors *could* perform in given situations, we examine what it would mean for system functions if they *did not* perform that work and assume that each function requires at least one person to perform it. The stress test was carried using a simple Python script together with the NetworkX library [18]. In pseudo-code the script was as follows:

```
START
      Load and parse WESTT[2] file format
      Construct networks
      Calculate network metrics
      Rank nodes by degree (n of edges)
      Loop start:
            Calculate network metrics
```

[2] The WESTT (Workload, Error, Situation Awareness, Time and Teamwork) application is a system for representing and analysis team-based work and is described in [16] and [17].

```
   Remove node (in     ascending or descending order of degree)
           Carry out clean-up
           Remove orphan nodes
           Report removal of orphaned task nodes
  Loop end
END
```

The outputs of this process are shown in Figure 10. Two types of node removal were used. In the ascending pattern, actors were removed in ascending order of degree (that is, the number of connections they have). Conversely, in the descending pattern actors with the highest degree were removed first. When task nodes were orphaned, that is, no actor was connected to them (and thus no actor was doing that work) they were also removed and this was recorded. Taken together, the ascending condition gives what might be a best-case pattern (the 'busiest' in connectivity terms removed first) for node removal and the descending condition a worst case. Other patterns of disruption would fall between these extremes (subject to combinatorial possibilities that could be probed by trying random permutations).

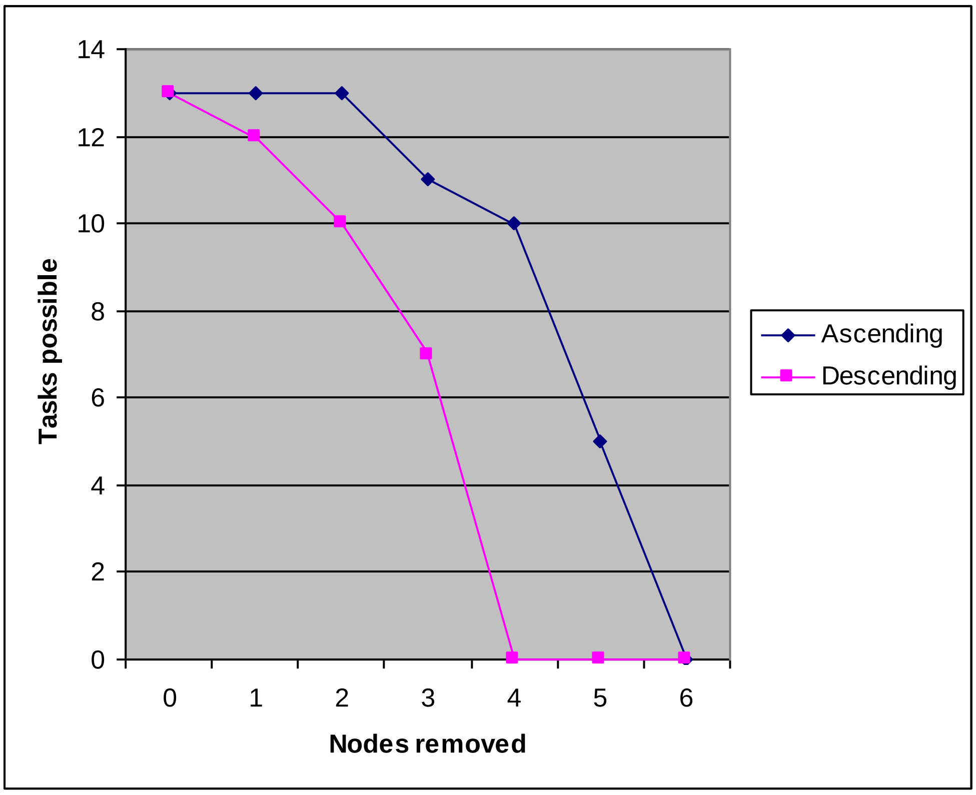


Figure 10. Graph of usecase stress testing.

The present example is arguably a little too small and straightforward to get the best out of this approach, but from the stress testing we see that the present design, as we have staffed it, puts great reliance on the availability of Ops and Command personnel without whose input things quickly deteriorate, with slightly less emphasis being put on Plans and Intel, and finally with JAG (and Recce, as we might expect) being relatively peripheral.

Because personnel are considered in terms of manning each activity, the usecase for both the workflow and distributed forms of the decision chain are the same. However, this does not take into account the workload that performing multiple actions at the same time would impose on staff; thus if we make the (simplistic) assumption that only one named person can carry out one action per epoch, and that where there is redundant capacity these individuals will be allocated to tasks they can perform, we end up with the slightly different usecase in Figure 11. This includes multiple Command, Intel and Ops staff to meet the potential for simultaneous demands on their time.

This perhaps illustrates one of the downside of flexible scheduling of operations; it is possible that for individuals that either they may face considerable amounts of workload simultaneously or else, a larger staff may be required to make sure ‘peak demand’ can be met. In contrast, workflow-based systems are less likely to suffer from this as opportunities for workload to vary are restricted by the structure of tasks.

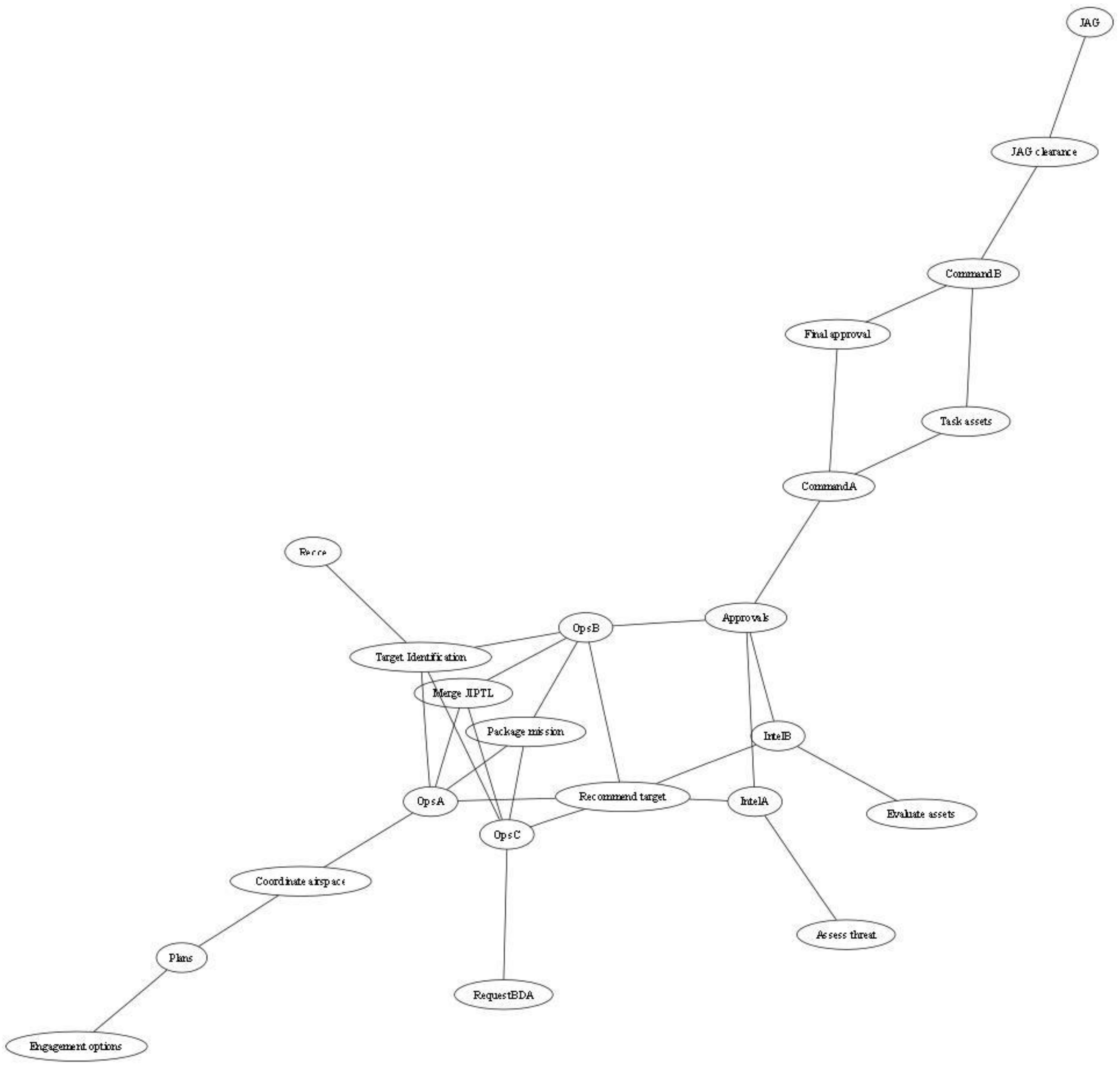


Figure 11. Distributed use-case

In future iterations of our EC-related work we intend to examine the issue of workload scheduling by staff in addition to consideration of the structure of activities.

4.0 CONCLUSION

The purpose of the present work was to pilot test two relatively novel forms of analysis for sociotechnical systems. In our view both EC and dynamic usecases have different strengths and weaknesses. These can be summarised thus;

- The event calculus is well-suited to modelling how systems perform over a period of time.
- By using different axiomisations via EC, it is possible to compare different paradigms one might wish to apply to sociotechnical systems on a like-with-like basis.
- There is a learning curve involved in forming adequate and appropriate propositions to capture the key elements of systems; however the output of this process is arguably much easier to understand than, for example, mathematical outputs, as logical statements translate easily into normal English.
- Furthermore, the rather involved process of constructing axioms has the useful side effect that it forces the analyst to clarify their understanding of the system in a formal manner; arguably if we lack the information to construct axioms, we lack a detailed understanding of the system. Thus there is a benefit “in the doing” in addition to the output itself.
- Constructing these models is time consuming. While not excessively so compared to comparable human factors methods, for larger and more complex systems than those considered here it might be necessary to write a computer program that can itself produce EC logic programs rather than as in the present case, writing each line by hand. This approach has been used previously in work applying the EC to natural language understanding that employed a parser [13].
- Usecases are useful for visualising the relationships between agents and tasks (and the relationships between agents via tasks) and make it clear ‘at a glance’ which agents are most immersed in a given set of activities (and thus perhaps, most critical) and, through considering the number of connections, which agents face the greatest potential variations in workload.

In terms of how the EC and usecases may fit in and around other analysis techniques in Human Factors and Systems Engineers, we suggest the overarching process from elicitation to measurement in Figure 12. Thus we begin with an elicitation and representation phase in which information about an existing system or specifications for a future system are collected. At this point, something like Hierarchical Task Analysis might take place which would also lead to the initial draft of an EC model. Then, through model finding (abduction in the EC), various forms the system could take are produced on the basis of the elicited facts. At this stage it may well be necessary to go back to elicit further information in order to “sanity check” outputs and also to tighten up any aspects of the axiomisation that appear unclear or open to question. Thus we see the construction of EC models as in part an interactive process where the models are ideally tested against reality as they are developed.

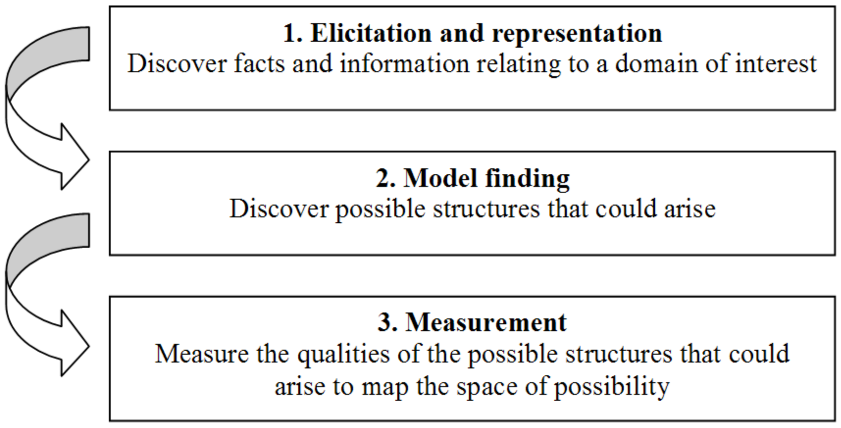

1. Elicitation and representation
Discover facts and information relating to a domain of interest
2. Model finding
Discover possible structures that could arise
3. Measurement
Measure the qualities of the possible structures that could arise to map the space of possibility

Figure 12. Suggested method for analytical prototyping

Once models have been found, these can then be submitted to simulation and measurement (this may include dynamic use cases, and also Petri Nets and other workload and error analysis techniques). Event calculus could also be present at this stage to see whether the system is compatible with different (and perhaps exceptional) narratives of events.

5.0 REFERENCES

[1] Kowalski, R. & Sergot, M. (1986). A logic-based calculus of events *New Generation Computing, 4,* 1, 67–95.

[2] Miller, R. & Shanahan, M. (2002). Some alternative formulations of the event calculus. In A.C. Kakas and F. Sadri (Eds.), *Computation and Logic: Essays in Honour of Robert Kowalski, Part II, Lecture notes in Computer Science*. Berlin: Springer, 452-490.

[3] Mueller, E. T. (2006). *Commonsense Reasoning*. New York, NY: Morgan Kaufman.

[4] Mueller, E. T. (2009). Automating commonsense reasoning using the event calculus. *Communications of the ACM, 52*, 1, 113-117.

[5] Yolum, P. & Singh, M. P. (2004). Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. *Artificial Intelligence, 42,* 1-3. 227-253.

[6] Shanahan, M. (2005). Perception as abduction: Turning sensor data into meaningful representation. *Cognitive Science, 29,* 103-134.

[7] Reason, J. T. (1990). *Human Error*. Cambridge, UK: Cambridge University Press

[8] Vicente, K. J. (1999). *Cognitive work analysis: Towards safe, productive and healthy computer-based work*. Mahwah, NJ: Lawrence Erlbaum Associates

[9] Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communications*. Cambridge, UK: Cambridge University Press

[10] Denecker, M., Van Belleghem, K., Duchatelet, G., Piessons, F., & De Schreye, D. (1996). A realistic experiment in knowledge representation in open event calculus: Protocol specification. *Proceedings of the Join International Conference and Symposium on Logic Programming (JICSLP'96)*, September 2-6, 1996, Bonn, Germany.

[11] Cicekli, N. K., & Yildirim, Y. (2000). Formalising workflows using the event calculus. In M.T. Ibrahim and J. Kung and N. Revell (Eds.) *Database and expert systems applications* (Lecture Notes in Computer Science, Vol. 1873, pp. 222-231). Berlin: Springer.

[12] Workflow Management Coalition (1996). *Terminology and Glossary (Technical Report WFMCTC-1011)*. Brussles: WMC.

[13] Mueller, E. T. (2002). Understanding script-based stories using commonsense reasoning. *Cognitive Systems Rsearch, 5* (4), 307-340.

[14] Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., Hammond, P. & Cory, H. T. (1986). The British Nationality Act as a logic program. *Communications of the ACM, 29,* 5, 370-386.

[15] A. K. S. Ng & J. Efstathiou, Structural robustness in complex networks, *NetSci International Workshop on Network Science*, Bloomington, IN, May 2006.

[16] Houghton, R. J., Baber, C., Cowton, M., Stanton, N., & Walker, G. (2008). WESTT (Workload, Error, Situational Awareness, Time and Teamwork): An analytical prototyping system for command and control. *Cognition, Technology and Work, 10,* 199-207.

[17] Baber, C., Houghton, R. J., & Cowton, M. (2004). WESTT: A Reconfigurable Human Factors Model for Network Enabled Capability. *NATO Technology and Research Organisation: Modelling and Simulation to Address NATO's New and Existing Military Requirements*, Germany, September 2004.

[18] A. A. Hagberg, D. A. Schult & and P. J. Swart, Exploring network structure, dynamics and functions with NetworkX, *Proceedings of the 7th Python In Science (SciPy2008)*, G. Varoquaux, T. Vaught and J. Millman (Eds.), Passadena, CA: pp. 11-15, August 2008.

ACKNOWLEDGEMENT

This work is supported by a grant from the Human Factors Integration Defence Technology Centre, part-funded by the Human Capability Domain of the UK Ministry of Defence Scientific Research Programme.